Installing on Ubuntu

Issues when installing CDAT on the Ubuntu distribution of GNU/Linux.

Introduction

In early Feb 2006 I did the following express install at the Unix command line:

```
$ sudo ./express_install /usr/local/cdat-4.0 --enable-ioapi --disable-opendap
```

of CDAT 4.0 on a four−processor Dell Poweredge 6300 running Ubuntu 5.10 Server with the symmetric multi−processing (smp) kernel. This works, but there are some fixes you have to apply along the way.

Explicitly set the default compiler

If you don't explicitly set the environment variables specifying which compilers to use, you may end up compiling different libraries with different compilers. Thus, in csh or tcsh, make sure you execute:

```
setenv CC gcc
setenv FC g77
setenv CXX g++
```

or in Bash:

```
export CC=gcc
export FC=g77
export CXX=g++
```

Do this before you run the express install.

But I want OpenDAP!

If you want OpenDAP then leave out the `--disable-opendap` keyword in the above `express_install` call. However, installation with OpenDAP often fails, and since I didn't need OpenDAP, I decided to try the install with it disabled.

If at first you don't succeed ...

Throughout the install process, different problems might come up (packages aren't found, things don't build, etc.). In general, if that happens, my advice is to just delete everything in the target directory and the source directory, unpack the `tar` file again, and reinstall. (The recommended way is to just clean up the build directory, but I like the brute force approach to cleaning.) Sometimes this is enough to fix the problem.

If reinstalling doesn't work, then you'll have to look at the log files. (Btw, the logs are in `exsrc/build` and are named `<package-name>.LOG`. You might also be interested in the `config.log` files which are located in the package sub−directories in `exsrc/build`.)

Ubuntu packages needed

If you have only the minimal Ubuntu install, there is a good chance that you won't be able to do a successful install of CDAT, since additional packages are needed. Here's the list of what I added (you can add them using Synaptic, Ubuntu's GUI−based package manager):

- bison
- byacc
- flex
- gawk
- g++
- libxt-dev
- tcsh
- tk8.4-dev
- xlibs-dev

Of these packages, xlibs-dev is the really key one (see, for instance, Alexis Zubrow's experience with installing on Debian, which is very similar to Ubuntu). You may be able to do a successful build without adding all of the others, but because I didn't have the time to try reinstalling by adding one package at a time, I don't know if all the above is necessary. But with them all, it works.

gplot

First, in order to compile gplot, make sure you have the libxt-dev and xlibs-dev libraries installed. It won't work without them.

The next parts are tricky. You have to edit the Makefile.Linux file in <CDAT_SRC_DIRECTORY>/exsrc/src/gplot, where you compile gplot. The problem is that the X Windows libraries, which are normally in /usr/X11R6/lib are in /usr/lib in Ubuntu. But it gets worse. A number of the libraries in /usr/lib are not found, not because they are not present, but because they are named by version number. As an example, do ls /usr/lib/libXp.*. You'll find there are only two files, libXp.so.6 and libXp.so.6.2.1. The libXp.so file is missing, which will cause the gplot compilation to break.

To prevent this, I created softlinks for all the "missing" libraries that are given in line 87 of the makefile:

```
MLIBS = -L/usr/X11R6/lib -lXp -lXpm -lXaw -lXmu -lXext -lXt -lX11
```

Thus, I created a libXp.so file by soft linking it to libXp.so.6.2.1, and so on for all the libraries specified in the line above that did not exist in /usr/lib. (Actually, to be more precise, I made a copy of all the libraries in /usr/lib, created the softlinks, and changed the -L flag in the makefile to that new directory. I did this because I didn't want to make changes to /usr/lib.

With these changes, gplot compiles fine, and you can following the rest of the instructions for installing gplot.